연결 해제 모드의 모빌리티 워크로드용 UCS의 RHOSP 구축 설명서

목차

소개

이 문서에서는 Cisco VPC-DI를 지원하기 위해 C220 M6 UCS Server에 RHOSP를 구축하기 위한 프레임워크에 대해 설명합니다.

사전 요구 사항

요구 사항

Cisco에서는 Red Hat OpenStack Platform(RHOSP)에 대한 지식이 있고 Red Hat Enterprise Linux(RHEL)에 대한 강력한 기술을 보유한 것을 권장합니다. 또한 가상화 및 네트워킹 개념에 대한 확고한 이해가 필요합니다.

사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

이 가이드에서는 확장성, 신뢰성 및 성능 최적화를 강조하면서 RHOSP와 UCS(Unified Computing System) 인프라의 통합에 대해 설명합니다.

모범 사례를 자세히 설명하고, Undercloud 및 Overcloud 아키텍처로 구성된 OpenStack TripleO를 배포하기 위해 가능한 스크립트 기반 자동화를 사용합니다.

이 구축 가이드를 사용하면 Cisco VPC-DI(Virtual Packet Core - Distributed Instance) 기반 VNF(Mobility Virtual Network Functions)를 지원하는 견고하고 효율적인 RHOSP 클라우드 인프라를 구축할 수 있습니다.

RHOSP 이해

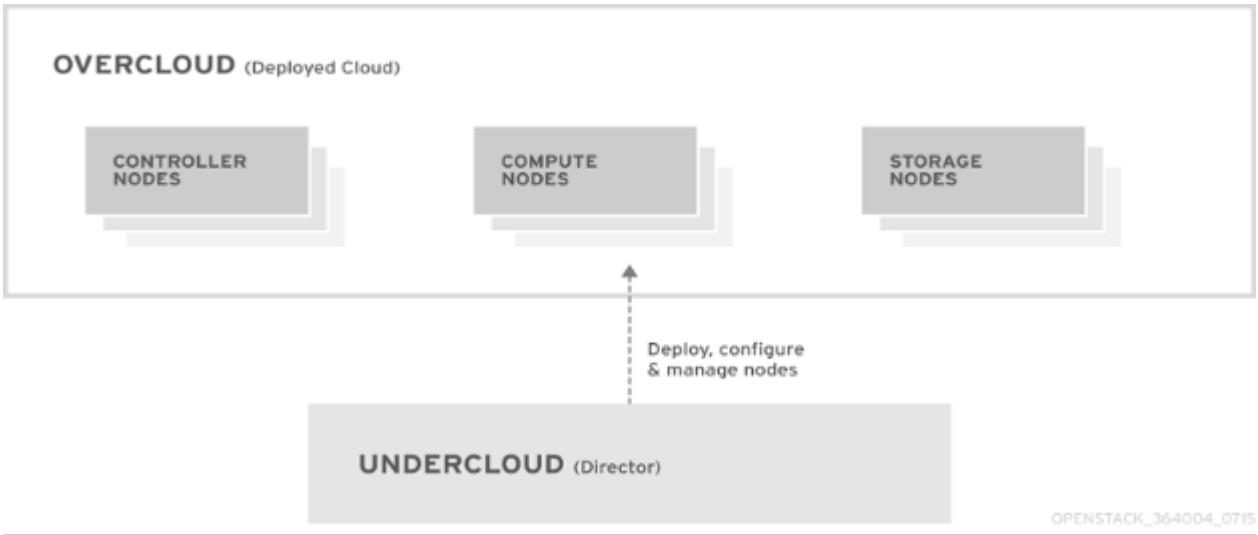
RHOSP는 오픈 소스 OpenStack 프로젝트를 기반으로 구축된 엔터프라이즈급 프라이빗 클라우드 솔루션으로서 Red Hat과 통합되고 지원됩니다. VM(Virtual Machine), 네트워킹 및 스토리지를 온디맨드 방식으로 IaaS(Infrastructure-as-a-Service)를 구축하고 관리할 수 있습니다.

HA(High Availability), 네트워크 기능 가상화, 맞춤형 구축 등의 기능을 제공합니다.

RHOSP 아키텍처

RHOSP는 주로 OpenStack TripleO 프로젝트를 기반으로 합니다. Openstack은 완벽한 RHOSP 환경을 설치 및 관리하기 위한 툴 세트 역할을 하는 Director를 사용합니다.

RHOSP는 확장 가능하고 유연한 클라우드 인프라를 제공하도록 설계되었습니다. 이 아키텍처는 두 가지 주요 구성 요소로 이루어져 있습니다. Undercloud와 Overcloud.



언더클라우드

Undercloud는 RHOSP 디렉터 도구 집합을 포함하는 기본 관리 노드입니다. OpenStack 환경(오버클라우드)을 구성하는 OpenStack 노드를 프로비저닝하고 관리하기 위한 구성 요소가 포함된 단일 시스템 OpenStack 설치입니다.

언더클라우드 구성 요소

Undercloud는 OpenStack 구성 요소를 기본 툴 세트로 사용합니다. 각 구성 요소는 언더클라우드의 개별 컨테이너 내에서 작동합니다.

- OpenStack ID(keystone) - 디렉터 구성 요소에 대한 인증 및 권한 부여 제공
- OpenStack 베어 메탈(아이러니한) - 베어 메탈 노드 관리
- OpenStack Networking(neutron) 및 Open vSwitch - 베어 메탈 노드를 위한 제어 네트워킹
- OpenStack Orchestration(임시 열) - 디렉터가 오버클라우드 이미지를 디스크에 쓴 후 노드의 오케스트레이션을 제공합니다.

오버클라우드

오버클라우드는 언더클라우드를 사용하여 생성된 RHOSP 환경입니다. 여기에는 고객이 만들고자 하는 OSP(OpenStack Platform) 환경을 기반으로 정의된 다양한 노드 역할이 포함됩니다.

컨트롤러

컨트롤러 노드는 OpenStack 환경을 위한 관리, 네트워킹 및 HA를 제공합니다. 권장 OpenStack 환경에는 HA 클러스터에 3개의 컨트롤러 노드가 함께 포함되어 있습니다.

컴퓨팅

컴퓨팅 노드는 OpenStack 환경을 위한 컴퓨팅 리소스를 제공합니다. 컴퓨팅 노드는 시간의 경과에 따른 네트워크 요구 사항에 따라 Scale-In/Scale-Out될 수 있습니다. 기본 컴퓨팅 노드에는 다음과 같은 구성 요소가 포함되어 있습니다.

- OpenStack Compute(nova)
- 커널 기반 VM(KVM)/빠른 에뮬레이터(QEMU)
- vSwitch 열기

스토리지

스토리지 노드는 OpenStack 환경을 위한 스토리지를 제공합니다.

참고: 고객 네트워크에서 언더클라우드/디렉터 및 오프라인 저장소(REPO)를 구축하는 방법에는 여러 가지가 있습니다. 베어 메탈 노드에 직접 구축하거나 KVM 하이퍼바이저 위에 VM으로 구축할 수 있습니다. 현재 구축 가이드에서는 Director UCS 서버가 KVM(하이퍼바이저)을 호스트하여 여러 VM을 상단에 구축합니다. RHOSP Director 노드 및 Offline-REPO 노드는 KVM Hypervisor에 VM으로 구축됩니다.

로컬 저장소(REPO 서버)

Redhat는 CDN(Content Delivery Network)에서 패키지를 다운로드하는 데 사용할 수 있는 reposync라는 유틸리티를 제공합니다. 특정 채널에서 모든 패키지를 다운로드하려면 시스템이 해당 채널에 가입되어 있어야 합니다. 시스템이 필수 채널에 가입되어 있지 않은 경우 reposync는 로컬 시스템에서 해당 패키지를 다운로드하고 동기화할 수 없습니다.

리포지토리는 확장명이 .repo로 끝나는 파일별로 /etc/yum.repos.d/path로 구성됩니다. 동일한 파일에 여러 저장소를 정의할 수 있습니다.

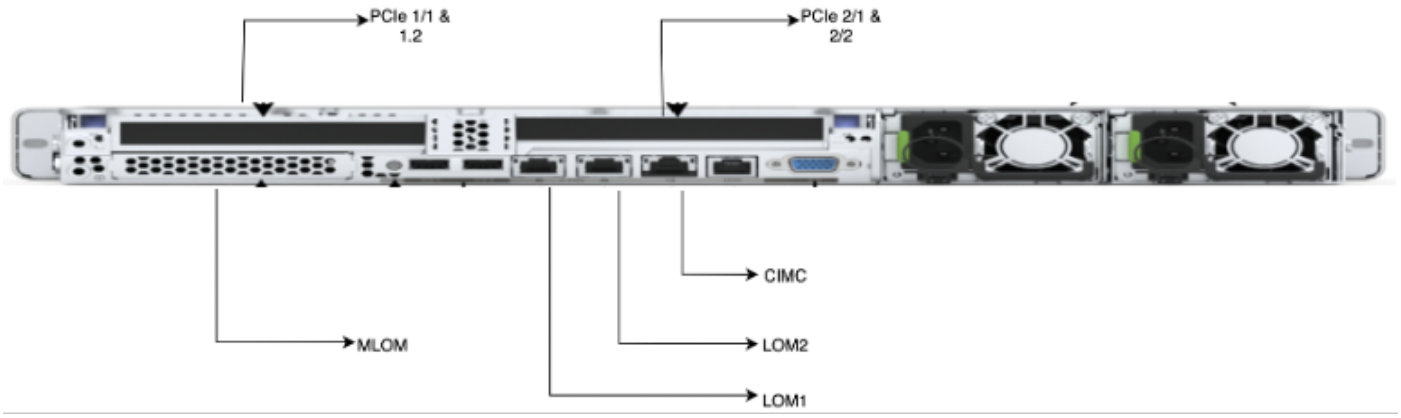
RHOSP 네트워킹

네트워킹 서비스(neutron)는 RHOSP의 SDN(Software-Defined Networking) 구성 요소입니다. RHOSP 네트워킹 서비스는 VM 인스턴스를 오가는 내부 및 외부 트래픽을 관리하고 라우팅, 세그멘테이션, DHCP, 메타데이터 등의 핵심 서비스를 제공합니다. 가상 네트워킹 기능과 스위치, 라우터, 포트 및 방화벽 관리를 위한 API를 제공합니다.

RHOSP Director는 OpenStack 서비스를 서로 다른 격리된 네트워크에 매핑합니다. 각 트래픽 유형을 전달하는 네트워크는 다음과 같습니다. Cisco CIMC(Integrated Management Controller), 프로비저닝, 내부 API, 스토리지 데이터, 스토리지 관리, 테넌트 및 외부(SSH(Secure Shell) 및 OAM(Operations, Administration, and Maintenance).

RHOSP 물리적 연결

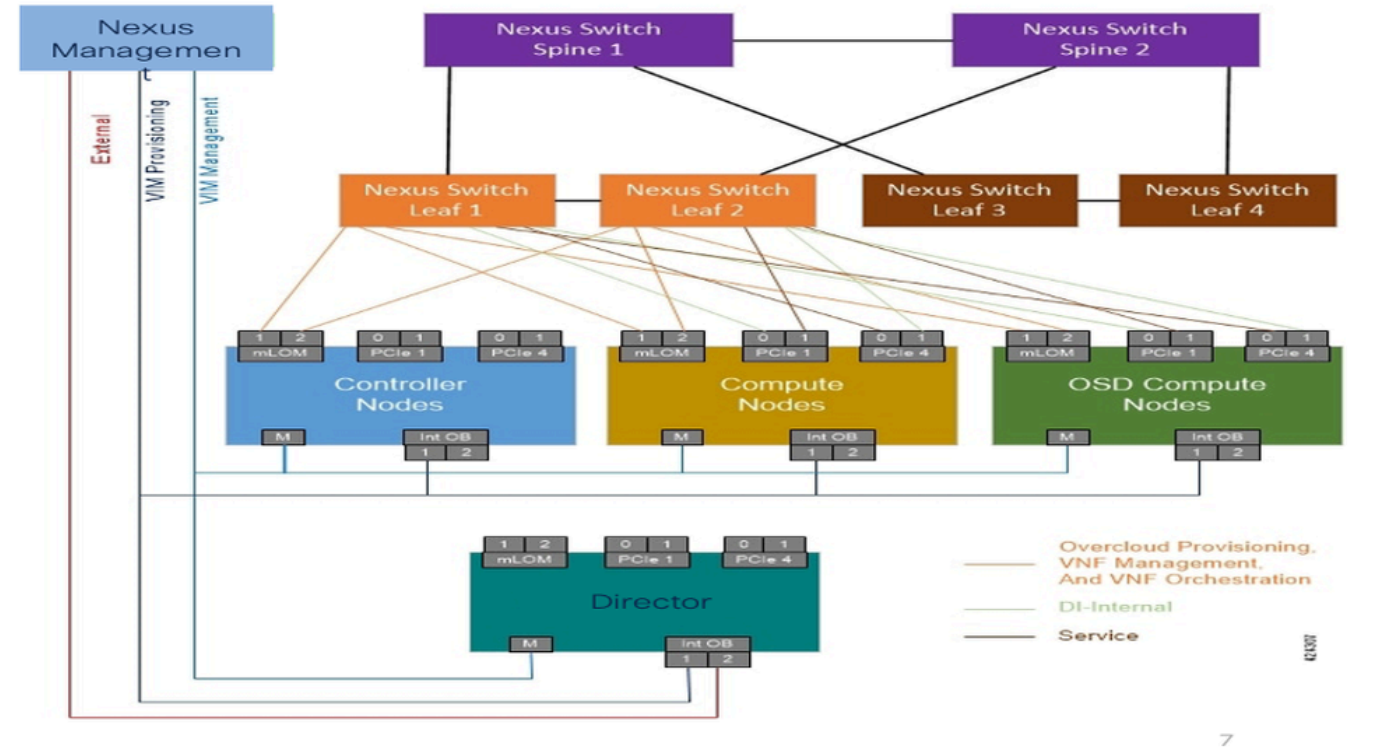
RHOSP 구축에서는 서로 다른 연결 목적을 위해 Cisco UCS C220 M6 서버의 서로 다른 물리적 포트를 사용합니다.



일련 번호	물리적 포트	세부사항
1.	CIMC	CIMC는 서버 프로비저닝 및 관리를 위한 OOB(Out of Band) 연결을 제공합니다.
2 .	SR-IOV(Single Root I/O Virtualization)/PCIe(Peripheral Component Interconnect Express)	PCIe NIC(Network Interface Card)는 VNF의 DI 내부 및 서비스 네트워크용 컴퓨팅 노드에 사용됩니다.

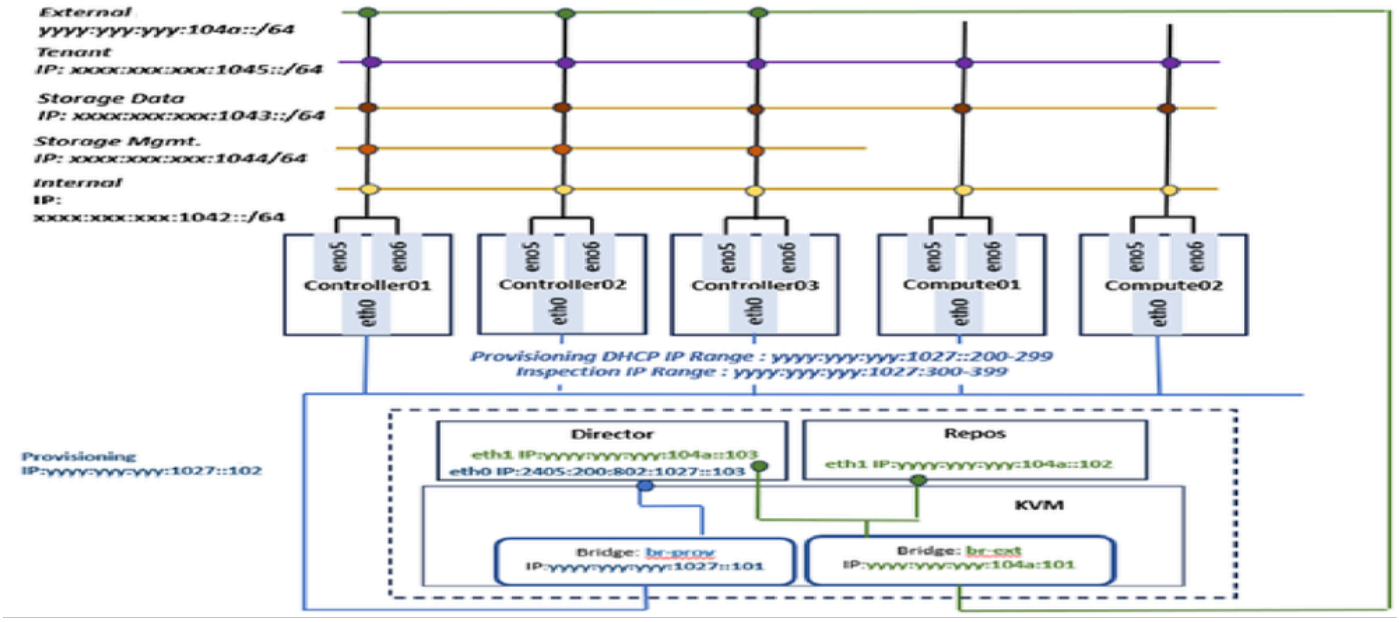
3 .	MLOM(Modular Lan on MotherBoard)	MLOM 포트는 본드로 구성됩니다. osp_external, osp_internal, osp_tenant, osp_external, osp_storage_data, osp_storage_mgmt는 내부 통신을 위해 MLOM 포트를 사용합니다.
4.	LOM(Lan on MotherBoard)	디렉터는 LOM1 및 LOM2 포트를 사용하는 반면, Computers 및 Controllers는 LOM1 포트만 사용합니다. LOM1은 모든 서버에 Openstack을 구축하거나 프로비저닝하는 데 사용됩니다. LOM2는 디렉터에서 OAM(External Network)으로 사용됩니다.

다이어그램에는 서버와의 물리적 연결이 나와 있습니다.



RHOSP 논리적 연결

RHOSP 네트워크에는 클라우드 내의 다양한 서비스를 제공하는 여러 서브넷이 있습니다.



- OSP_CIMC:

CIMC는 모든 UCS 서버의 관리를 제어하는 IPMI(Intelligent Programming Management Interface)입니다. 이 CIMC 네트워크는 모든 UCS 서버의 독립형 CIMC 포트에 구성됩니다.

- OSP_프로비저닝:

이 네트워크는 오버클라우드 구축 중에 컴퓨팅 및 컨트롤러 서버의 프로비저닝 및 PXE(Preboot Execution Environment) 부팅 관리와 DHCP IP 가져오기를 담당합니다. 프로비저닝 네트워크는 간소화 및 호환성을 위해 모든 UCS 서버의 LOM1 포트에서 네이티브 VLAN으로 구성됩니다. 이 프로비저닝 네트워크는 모든 서버에 클라우드를 구축하는 역할을 담당합니다.

디렉터 서버의 가상화로 인해 디렉터 VM이 다른 서버와 통신할 수 있도록 KVM에 브리지 네트워크를 생성해야 했습니다.

- OSP_내부:

내부 API 네트워크는 중성자, 노바, 키스톤 등과 같은 OpenStack 서비스 간의 통신에 사용됩니다.

OSP_Internal 네트워크는 컨트롤러 및 컴퓨팅 노드의 결합된 MLOM 포트에 구성됩니다.

- OSP_테넌트:

테넌트 네트워크는 VNF 관리를 위해 클라우드 프로젝트 내에서 기본적으로 생성됩니다. 현재 설정에서는 VNF 구축을 위해 단일 Openstack 프로젝트만 생성됩니다.

OSP_Tenant 네트워크는 컨트롤러 및 컴퓨팅 노드의 결합된 MLOM 포트에 구성됩니다.

- OSP_외부:

외부 네트워크는 모든 외부 액세스(예: SSH) 및 API 네트워크에 사용됩니다.

OSP_External 네트워크는 디렉터 노드의 LOM2 포트와 컨트롤러 및 컴퓨팅 노드의 결합된 MLOM 포트에 구성됩니다.

- OSP_스토리지_데이터:

OSP_Storage 네트워크는 스토리지 액세스와 관련된 모든 작업에 사용됩니다. 이는 스토리지에 액세스해야 하는 CEPH 서비스와 VNF 간의 통신에 필요합니다. 컨트롤러, 컴퓨팅 노드 및 CEPH에서 사용됩니다.

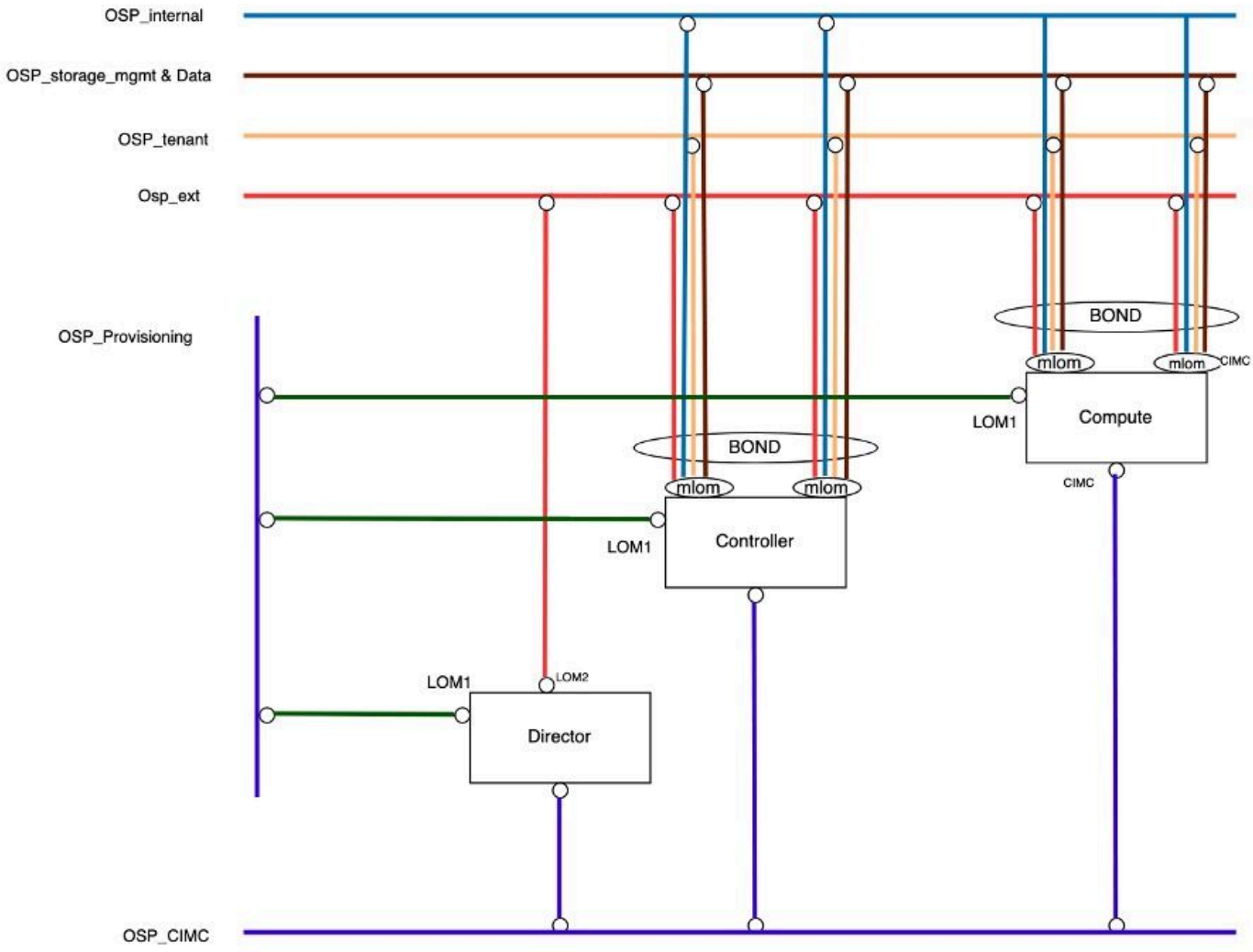
OSP_Storage_Data 네트워크는 컨트롤러 및 컴퓨팅 노드의 결합된 MLOM 포트에 구성됩니다.

- OSP_스토리지_관리:

OpenStack 개체 저장소는 이 네트워크를 사용하여 컨트롤러-컴퓨팅 노드 간에 형성된 저장소 클러스터의 참여 복제 노드 간에 데이터 개체를 동기화합니다.

OSP_Storage_Mgmt 네트워크는 컨트롤러 및 컴퓨팅 노드의 결합된 MLOM 포트에 구성됩니다.

이 다이어그램은 언더클라우드 논리 네트워크가 RHOSP 클러스터의 각 노드 유형에 연결되는 방식을 보여줍니다.



하드웨어 매개변수 조정

- 설계에 따라 IP가 활성화되고 올바른 관리 주소로 구성되었는지 확인합니다.
- 모든 CIMC 로그인 액세스에 대해 admin 사용자 및 비밀번호가 동일해야 합니다.
- 부트스트랩(부팅) 모드가 UEFI(Unified Extensible Firmware Interface)로 설정되어 있는지 확인합니다.
- 권장 매개변수로 BIOS 설정을 지정해야 합니다.
- Boot-Order(부팅 순서)를 다음으로 설정:
    1. LOM-PXE(eno1)
    2. HDD(부팅 하드 디스크 드라이브)

- LOM1(eno1)의 mac-address를 설정합니다.
- 모든 UCS/CIMC 서버가 최신 권장 펌웨어 버전으로 업그레이드되었는지 확인합니다.

하이퍼바이저-KVM 설치 및 네트워크 생성

고객 네트워크에 Undercloud/Director 및 Offline REPO를 구축하는 방법에는 여러 가지가 있습니다. 베어 메탈 노드에 직접 구축하거나 KVM 하이퍼바이저에서 실행되는 VM으로 구축할 수 있습니다. 현재 구축 가이드에서 Director UCS 서버는 KVM 하이퍼바이저를 호스팅하도록 구성되어 여러 VM을 쉽게 생성할 수 있습니다. RHOSP Director 노드 및 오프라인 REPO 노드는 이 KVM 하이퍼바이저에 VM으로 구축됩니다.

참고: KVM 하이퍼바이저를 구축하려면 표준 RHEL KVM 설치 단계를 준수해야 합니다.

- KVM이 가동되면 호스트 이름을 업데이트합니다.
  hostnamectl set-hostname <hostname> —static

- 외부 및 프로비저닝 브리지를 구성하고 인터페이스를 바인딩합니다.

br-prov: eth0

br-ext: eth1

이러한 브리지는 NMTUI(Network Manager Text User Interface) GUI를 통해 생성해야 합니다.

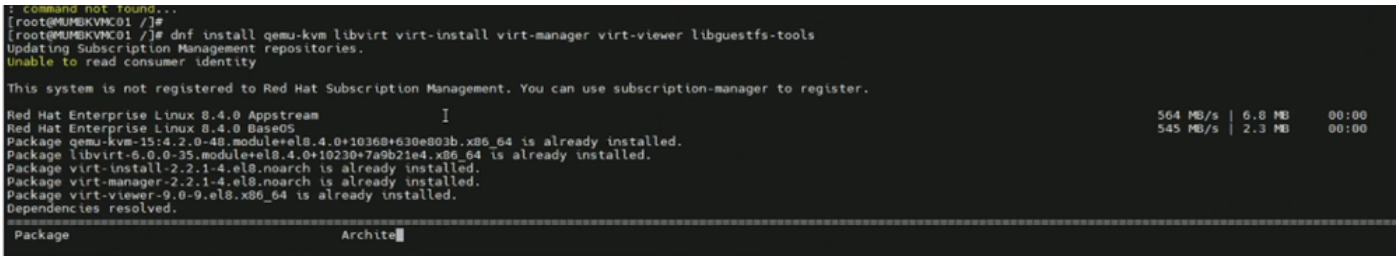
- 기본적으로 eno1 및 eno2 포트는 KVM에서 물리적 LOM1 및 LOM2 포트에 대해 생성되고 CIMC 로깅의 MAC 주소를 사용하여 동일한 포트를 교차 검증합니다.
- 올바른 MAC 주소를 매핑하여 브리지 네트워크를 만들고 여기에 슬레이브 포트를 추가합니다.
- 브리지 네트워크 생성 후 KVM에서 프로비저닝 및 외부 게이트웨이에 연결할 수 있는지 확인합니다.

REPO 및 디렉터 VM 생성

REPO 및 디렉터 VM 생성을 위한 사전 요구 사항

```
# dnf install qemu-kvm libvirt virt-install virt-manager virt-viewer libguestfs-tools
```

- REPO 및 디렉터 VM 생성을 위해 KVM에 필요한 패키지를 설치합니다.



- KVM에 필요한 디렉터리를 만듭니다.

mkdir /data # mkdir /data/offlineRepos # mkdir /data/isoImages # mkdir /data/qcow2Images # mkdir /data/images

- 파일을 디렉토리에 복사합니다.

```
# scp -r root@[remote-IP]:/root/rhel-8.4-x86_64-dvd.iso /data/isoImages/
# scp -r root@[remote-IP]:/root/offlineRepos/RHEL8.4 /data/offlineRepos/
# scp -r root@[remote-IP]:/etc/yum.repos.d/offlinedvd.repo /etc/yum.repos.d/
# scp -r root@[remote-IP]:/root/rhel-8.4-x86_64-kvm.qcow2 /data/qcow2Images/
# scp -r root@[remote-IP]:/root/OSREPO_RHEL_84.qcow2 /data/images/
# scp -r root@[remote-IP]:/root/OSREPO_DIRECTOR_84.qcow2 /data/images/
```

- ISO를 /mnt/iso에 마운트합니다.

```
# mount -t iso9660 -o loop /data/isoImages/rhel-8.4-x86_64-dvd.iso /mnt/iso
```

- /etc/yum.repos.d 경로에 REPO 파일을 생성합니다.

```
# cat /etc/yum.repos.d/offlinedvd.repo

[RHEL8.4_Appstream]
name=Red Hat Enterprise Linux 8.4.0 Appstream
mediaid=None
metadata_expire=-1
gpgcheck=0
enabled=1
baseurl=file:///data/offlineRepos/RHEL8.4/AppStream/

[RHEL8.4_BaseOS]
name=Red Hat Enterprise Linux 8.4.0 BaseOS
mediaid=None
metadata_expire=-1
gpgcheck=0
enabled=1
```

```
baseurl=file:///data/offlineRepos/RHEL8.4/BaseOS/
```

- 리포리스트를 확인하고 Appstream 및 Baseos REPO가 매핑되었는지 확인합니다.

```
# dnf repolist
```

```
[root@MUMBKVMC01 /]#
[root@MUMBKVMC01 /]#
[root@MUMBKVMC01 /]# dnf repolist
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

repo id                                                                 repo name
RHEL8.4_Appstream                                                       Red Hat Enterprise Linux 8.4.0 Appstream
RHEL8.4_BaseOS                                                          Red Hat Enterprise Linux 8.4.0 BaseOS
[root@MUMBKVMC01 /]#
[root@MUMBKVMC01 /]#
```

REPO VM 생성

```
$ cd /var/lib/libvirt/images/
$ export LIBGUESTFS_BACKEND=direct
$ virt-customize -a /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 --root-password password:Cisco@12
```

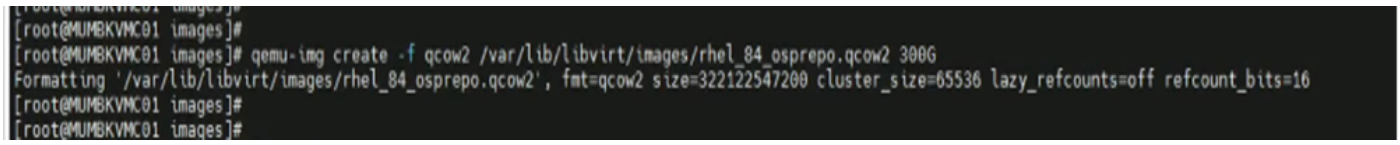
```
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# virt-customize -a /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 --root-password password:Cisco@123
[   0.0] Examining the guest ...
[   6.6] Setting a random seed
[   6.7] Setting the machine ID in /etc/machine-id
[   6.7] Setting passwords
[   7.7] Finishing off
[root@MUMBKVMC01 images]#
```

```
$ virt-filesystems --long -h --all -a /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2
```

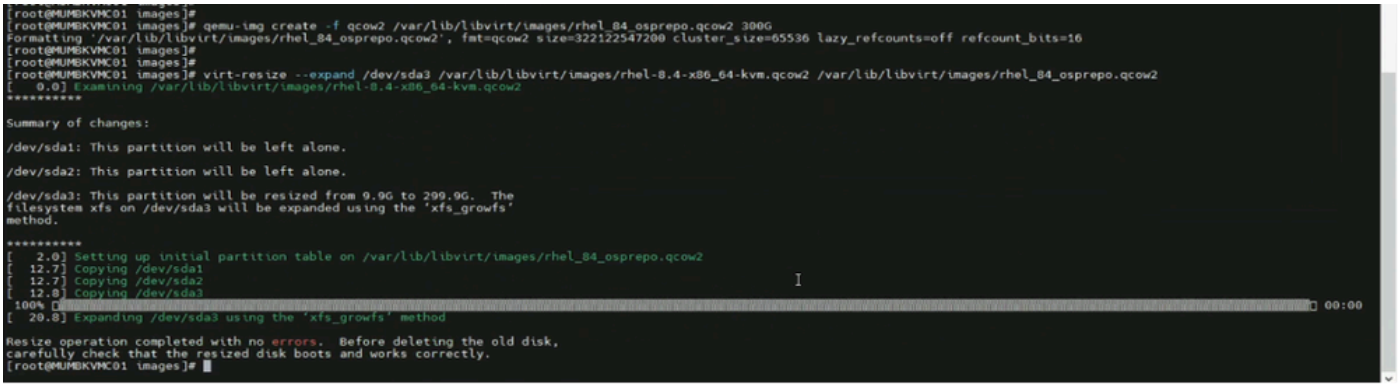
```
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# virt-filesystems --long -h --all -a /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2
Name       Type        VFS      Label  MBR  Size  Parent
/dev/sda1  filesystem  unknown  -      -    1.0M  -
/dev/sda2  filesystem  vfat     -      -    100M  -
/dev/sda3  filesystem  xfs      root   -    9.9G  -
/dev/sda1  partition   -        -      -    1.0M  /dev/sda
/dev/sda2  partition   -        -      -    100M  /dev/sda
/dev/sda3  partition   -        -      -    9.9G  /dev/sda
/dev/sda   device      -        -      -    10G   -
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]#
```

- REPO 서버의 이미지를 만듭니다.

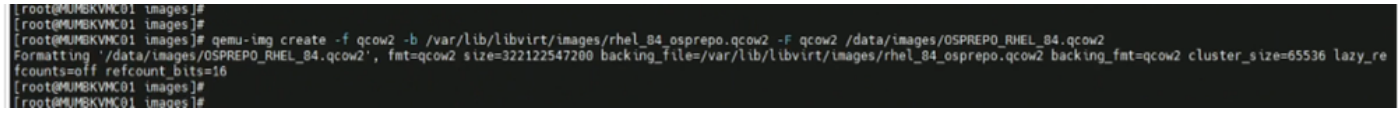
$ qemu-img create -f qcow2 /var/lib/libvirt/images/rhel_84_osprepo.qcow2 500G



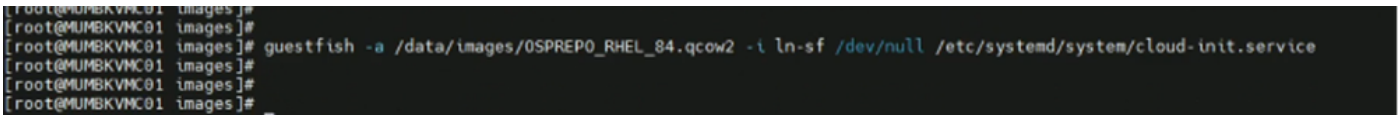
$ virt-resize --expand /dev/sda3 /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 /var/lib/libvirt/ima



$ qemu-img create -f qcow2 -b /var/lib/libvirt/images/rhel_84_osprepo.qcow2 -F qcow2 /data/images/OSPRE



$ guestfish -a /data/images/OSPREPO_RHEL_84.qcow2 -i ln-sf /dev/null /etc/systemd/system/cloud-init.serv



$ osinfo-query os | grep rhel8

```
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# osinfo-query os | grep rhel8

** (osinfo-query:49279): WARNING **: 17:12:33.856: Entity http://pcisig.com/pci/1B36/0100 referenced but not defined
 rhel8-unknown        | Red Hat Enterprise Linux 8 Unknown                | 8-unknown | http://redhat.com/rhel/8-unknown
 rhel8.0              | Red Hat Enterprise Linux 8.0                      | 8.0       | http://redhat.com/rhel/8.0
 rhel8.1              | Red Hat Enterprise Linux 8.1                      | 8.1       | http://redhat.com/rhel/8.1
 rhel8.2              | Red Hat Enterprise Linux 8.2                      | 8.2       | http://redhat.com/rhel/8.2
 rhel8.3              | Red Hat Enterprise Linux 8.3                      | 8.3       | http://redhat.com/rhel/8.3
 rhel8.4              | Red Hat Enterprise Linux 8.4                      | 8.4       | http://redhat.com/rhel/8.4
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]#
```

```
$ virt-install --cpu host --memory 32768 --vcpus 16 --os-variant rhel8.4 --disk path=/data/images/OSPRE
```

```
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# virt-install --cpu host --memory 32768 --vcpus 16 --os-variant rhel8.4 --disk path=/data/images/OSPREPO_RHEL_84.qcow2,device=disk,bus=virtio,format=qcow2
--import --noautoconsole --vnc --network bridge:br-ext --name OSPREPO_RHEL_84

** (process:49296): WARNING **: 17:13:15.813: Entity http://pcisig.com/pci/1B36/0100 referenced but not defined

Starting install...
Domain creation completed.
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]#
```

```
$ virsh list --all
```

```
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# virsh list --all
 Id    Name                           State
----------------------------------------------------
 1     OSPREPO_RHEL_84                running

[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# virsh list --all
 Id    Name                           State
----------------------------------------------------
 1     OSPREPO_RHEL_84                running

[root@MUMBKVMC01 images]#
```

디렉터 VM 생성

- 디렉터 서버의 이미지를 만듭니다.

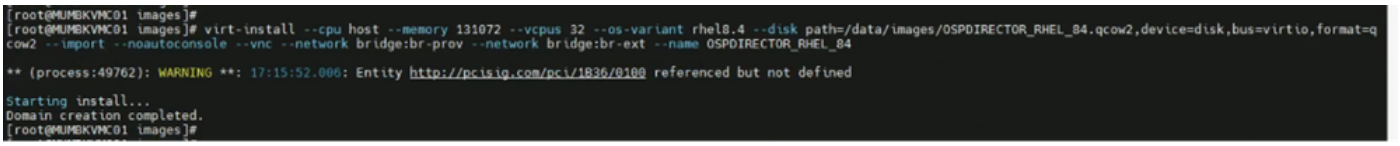
```
# qemu-img create -f qcow2 /var/lib/libvirt/images/rhel_84_ospdirector.qcow2 500G
```

```
[root@MUMBKVMC01 images]#
[root@MUMBKVMC01 images]# qemu-img create -f qcow2 /var/lib/libvirt/images/rhel_84_ospdirector.qcow2 200G
Formatting '/var/lib/libvirt/images/rhel_84_ospdirector.qcow2', fmt=qcow2 size=214748364800 cluster_size=65536 lazy_refcounts=off refcount_bits=16
[root@MUMBKVMC01 images]#
```

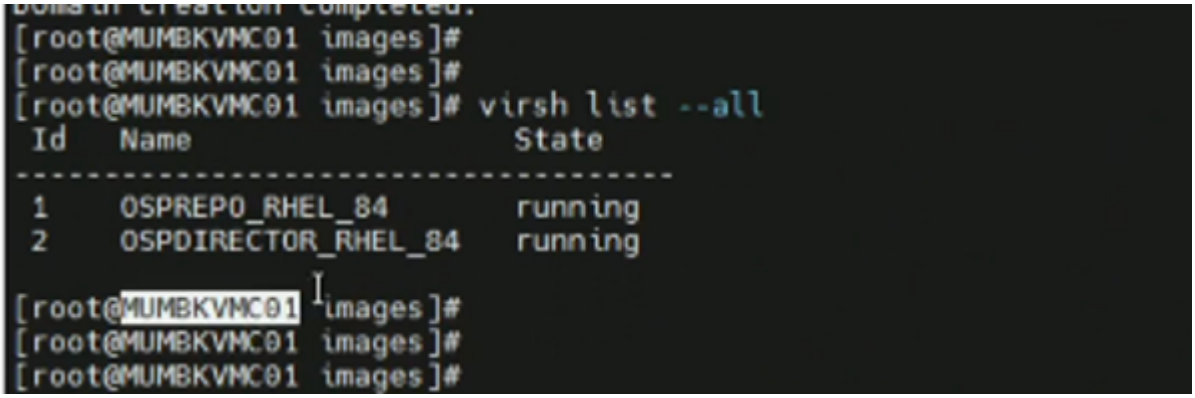
```
# virt-resize --expand /dev/sda3 /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 /var/lib/libvirt/ima
```

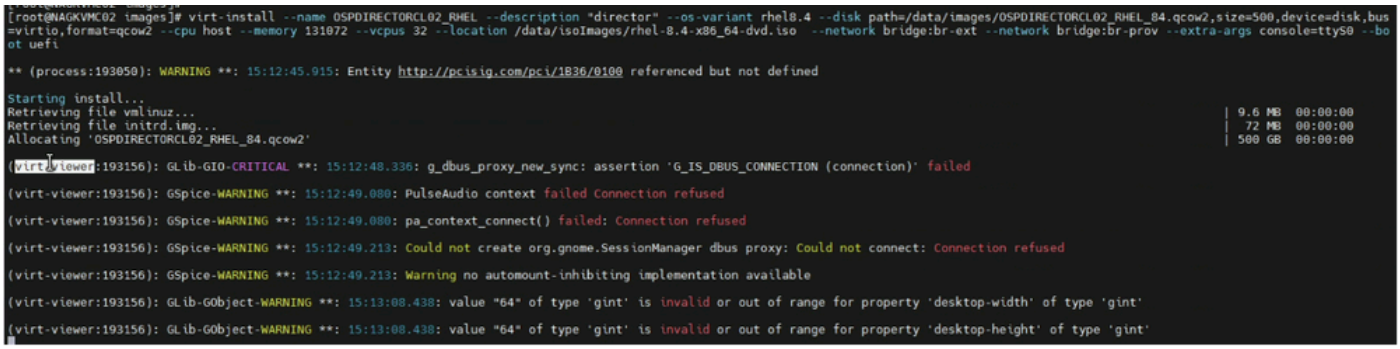

```
# qemu-img create -f qcow2 -b /var/lib/libvirt/images/rhel_84_ospdirector.qcow2 -F qcow2 /data/images/O
```


```
# guestfish -a /data/images/OSPDIRECTOR_RHEL_84.qcow2 -i ln-sf /dev/null /etc/systemd/system/cloud-init
# virt-install --cpu host --memory 131072 --vcpus 32 --os-variant rhel8.4 --disk path=/data/images/OSPD
```




```
# virsh list --all
```

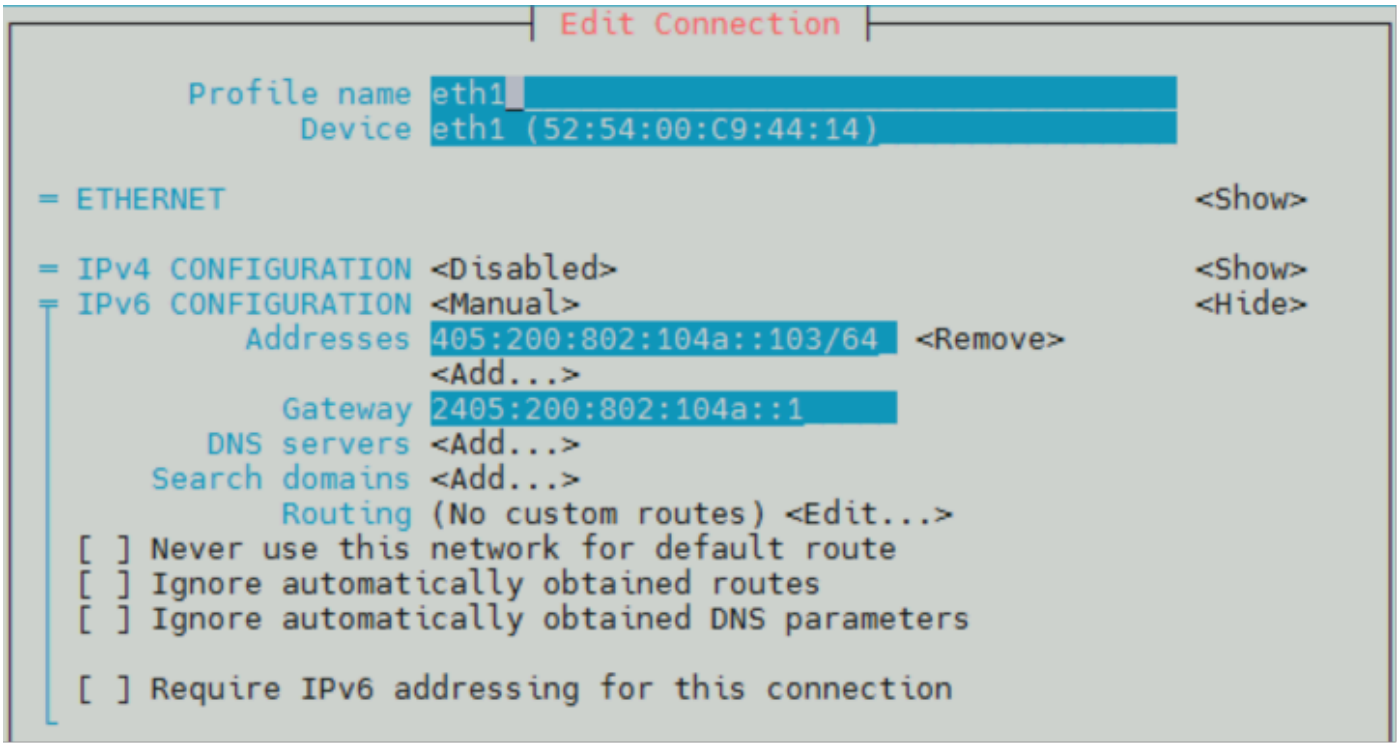




- 새 창을 열고 VM의 콘솔 로그인을 수행합니다.

```
# virsh list –all
# virsh console <domain-id>
```

- Post Director VM 구축에서는 두 개의 eth 인터페이스 eth0 및 eth1이 생성됩니다.
  - eth0은 OSP_provisioning 네트워킹에 사용되며, br-ctlplane 네트워크는 Undercloud 구축 중에 생성됩니다.
  - eth1은 SSH 액세스를 위한 외부 연결에 사용됩니다. 따라서 OSP_external 서브넷 IP 주소가 할당됩니다.



로컬 오프라인 REPO 생성

REPO Server는 Redhat CDN에 등록해야 하며 구축에 필요한 RHOSP 16.2의 사용 가능한 모든 패키지의 저장소가 있어야 합니다. RHEL RPM 패키지 및 RHOSP 컨테이너 이미지는 프록시를 사용하여 REPO VM에 다운로드해야 합니다.

RHOSP 클라우드 구축

RHOSP 16.2는 자동화를 통해 고객 네트워크에 구축됩니다. 실행 가능한 스크립트를 사용하여 Undercloud 및 Overcloud 구축을 자동화합니다.

사전 요구 사항

실제 클라우드 구축을 시작하기 전에 따라야 할 단계:

1. OSP_EXT 네트워크의 로컬 REPO VM 및 Director VM과 KVM 노드의 연결을 확인합니다.
2. 모든 가능한 스크립트 및 자동화 스크립트가 지정된 폴더의 KVM 호스트에 업로드되었는지 확인합니다.
3. 이름이 'cisco' 및 'automation'인 폴더를 만들고 실행 가능한 스크립트의 대상을 배치합니다.

```
# cd /home
# mkdir cisco
# cd /home/cisco
# mkdir automation
# cd /home/cisco/automation
```

tarball은 다음과 같은 세 가지 폴더 디렉토리 구조로 구성됩니다.

- 스크립트: 베어 메탈 노드 컨피그레이션에 사용되는 셸 스크립트로 구성됩니다.
- rpm: 인텔 이더넷 연결(ICE) 드라이버, 인텔 드라이버 및 Red-hat Package Manager(RPM) 패키지로 구성되어 있습니다.
- 가능: 입력 vars 파일, 배포용 yaml 파일 및 jinja-templates로 구성됩니다.

4. sshpass 패키지를 설치합니다. sshpass는 비대화형 ssh에 비밀번호를 제공하는 데 사용되는 명령줄 유틸리티입니다. 수동 비밀번호 입력이 불가능한 스크립팅 또는 자동화 시나리오에서 주로 사용됩니다.

- 인터넷/기존 서버에서 sshpass.tar.gz 패키지를 다운로드합니다.
- GCC(GNU Compiler Collection) 패키지를 설치합니다.

```
# yum install gcc
```

- 패키지 만들기를 설치합니다.

```
# yum install make
```

- sshpass 패키지의 압축을 풀고 SSH pass를 설치합니다.

```
# tar -xvzf sshpass.tar.gz
# cd sshpass-1.10/
# ./configure
# sudo make install
# sshpass -V
```

5. 디렉터 설치 프로세스에서는 루트가 아닌 사용자가 명령을 실행해야 합니다. '스택' 사용자는 Director VM에서 sudo 액세스로 생성해야 합니다.

```
# useradd stack
# passwd stack

Disable password requirements for the ‘stack’ user when using sudo.
# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
# chmod 0440 /etc/sudoers.d/stack
```

6. rootCA.crt 파일을 REPO 서버에서 지정된 경로의 Director VM 및 KVM에 복사합니다. 또한 신뢰 목록에서 REPO VM 인증서를 업데이트합니다.

```
# /etc/pki/ca-trust/source/anchors
# update ca-trust
```

7. /etc/hosts 파일의 Director VM 및 KVM에서 로컬 REPO-server 호스트 이름 세부 정보를 업데이트합니다.

8. KVM 및 Director VM에서 python, ansible 등과 같은 추가 패키지를 설치하여 ansible 자동화 스크립트를 실행합니다.

```
# dnf install python3 python3-devel ansible httpd -y
# update-alternatives --set python /usr/bin/python3
```

9. 클라우드 구축 중에 프로비저닝을 활성화하려면 Director의 프로비저닝 네트워크에서 CIMC 서브넷에 연결할 수 있어야 합니다. 필요한 경우 동일한 경로에 대한 고정 경로를 추가합니다.

```
# ip -6 route add <CIMC Subnet> via <Provisioning Subnet>
```

10. KVM 및 Director VM에서 /Ansible 폴더에 호스트 파일을 생성하고 필요에 따라 스택별 세부 정보를 추가합니다.

```
[ospd]
# <PODNAME> ansible_host=<OSPD IP> ansible_ssh_user=stack ansible_ssh_pass='<STACKPASSWD>' ansible_ssh_

<podname> - Stack Name of the Cloud.
<OSPD IP> - Baremetal OSPD Node IP Address
<STACKPASSWD> - OSPD Node password for ‘stack’ user
```

11. 모든 실행 가능한 플레이북과 입력 파일을 Director VM의 /home/stack 폴더에 보관합니다.

입력 파일 업데이트

클라우드 구축을 위해 준비해야 하는 고객 네트워크 관련 세부 사항으로 구성된 입력 변수 파일이 있습니다.

경로: /home/cisco/automation/ansible/podvars

파일 이름: <stack-name>_vars.yml

사이트별 IP 계획/하위 레벨 설계 문서에 따라 강조 표시된 매개변수를 업데이트합니다.

참고: 더미 IP 주소는 표시 목적으로만 사용됩니다.

<#root>

```
# ############################
# XR21 Specific Variables
# ############################

# ==========================
# Common Variables
# ==========================
```

```
# UCS hardware type: 'm4/m5/m6'
hardware: m6

# Platform type: 'epc/pcrf'
platform: epc

# RHEL version
rhel: { version: 84, tag: 8.4 }

# Openstack version
osp: { version: 16, major: 2 }

# Container version
container: { tag: 16.2, tools: 3.0 }

# Overcloud stack name

stack_name: '



    '




# OSPD full hostname

fqdn_hostname: '



    .epdg.ap.hamb.a6.cloud.com’



# OSPD host login
ospd_host: { ip:

'2405:XXXX:089:1054::11'

, username: 'stack', password: '*******' }

# OSPD cimc login
ospd_cimc: { ip:

'2405:XXXX:089:1054::11'
```

```
, username: 'admin', password: '********' }

# CIMC username and pasword must be same across all Overcloud nodes
cimc: { username: 'admin', password: '********', ip_pool: '2405:XXXX:089:1055::/64' }

# Undercloud-Overcloud provision
internal_network: {
   ip_type: 'v6',
   local_interface: 'eth0',

   local_ip: ‘2405:XXXX:089:1041::103',



   undercloud_public_host: ‘2405:XXXX:089:1041::105',



   undercloud_admin_host: ‘2405:XXXX:089:1041::104',



   cidr: ‘2405:XXXX:089:1041::/64',



   dhcp_start: ‘2405:XXXX:089:1041::200',



   dhcp_end: ‘2405:XXXX:089:1041::299',



   gateway: ‘2405:XXXX:089:1041::199',



#   nexthop: ‘2405:XXXX:089:1041::1',



   inspection_iprange_start: ‘2405:XXXX:089:1041::300',



   inspection_iprange_end: ‘2405:XXXX:089:1041::399',



}
```

```
# DNS
dns_ips: [ '2405:YYYY:a10:f100::1' ]
dns_search_domains: [ 'cloud.com' ]

# NTP
ntp_ips: [ '2405:YYYY:801:700::afa', '2405:YYYY:801:700::afb' ]

# Deployment type: 'offline/online'
repos: { rhel: 'offline', container: 'offline' }

# Offline details if repos is 'offline'
offline: {
    environment: 'v01_00',
    deliverymedia: '/home/stack/deliverymedia/'
}

# Satellite details if repos is 'online'
satellite: {
    fqdn_name: 'rh-satellite2.mitg-bxb300.cisco.com',
    ip: '10.XX.XX.XX',
    org: 'MITG',
    user: 'admin',
    password: '*******',
    environment: 'production',
    activation_key: 'ak-rhel{{rhel.version}}-osp{{osp.version}}{{osp.major}}',
    repos_file: 'rhel{{rhel.version}}osp{{osp.version}}{{osp.major}}.yaml'
}

# Offline container registry details
offline_registry: {
    ip: '2405:XXXX:089:1055::100',
    name: '

     .

      ',

    port: '5000',
    container_tag: '16.2.6',
    user: 'ciscoadmin',
    password: '******'
}

# Custom cloud domain details
domain_name: {
    domain: '
```

```
     ',


    cloudshortname: 'nl'
}

# Container images namespace
container_namespace: 'mitg-{{satellite.environment}}-cv-rhel{{rhel.version}}-osp{{osp.version}}{{osp.ma

# List of cimc IPs

ctrl_cimc_ip:


  - 2405:XXXX:YYYY:1036::12


  - 2405:XXXX:YYYY:1036::13


  - 2405:XXXX:YYYY:1036::14


osdc_cimc_ip:


cmpt_cimc_ip:


  - 2405:XXXX:YYYY:1037::17


  - 2405:XXXX:YYYY:1038::18


  - 2405:XXXX:YYYY:1038::19
```

```
  - 2405:XXXX:YYYY:1038::20



mgmt_cimc_ip:



  - 2405:XXXX:YYYY:1051::15



  - 2405:XXXX:YYYY:1051::16



# ============================
# Hardware Specific Variables
# ============================

# Isolcpu for cpu pinning
isolcpus: { osdc: '4-31,36-63', cmpt: '2-31,34-63', mgmt: '2-31,34-63' }

# Hugepages in 1G Pages
hugepages: { osdc: 428, cmpt: 448, mgmt: 448 }

# Reserved host memory in MB
reserved_host_memory: { osdc: 84000, cmpt: 64000, mgmt: 64000 }

# Number of VFs per SR-IOV port
sriov_vfs_per_port: 16

# List of SR-IOV ports
sriov_port_list: [ens1f0, ens1f1, ens9f0, ens9f1]

# List of OVS bonding interface
ovs_bond_interface: [eno5, eno6]

# Physical networks
physical_network: [phys_pcie1_0, phys_pcie1_1, phys_pcie2_0, phys_pcie2_1]

# Boot disk size
boot_disk_mb_size: { ctrl: 761985, osdc: 761985, cmpt: 761985, mgmt: 1524925 }

# Boot disk PD slot number
boot_disk_pd_slot: { ctrl: [1,2], osdc: [1,2], cmpt: [1,2], mgmt: [1,2] }

# Boot disk VD slot number
boot_disk_vd_slot: { ctrl: 237, osdc: 235, cmpt: 239, mgmt: 239 }

# Storage backend 'swift' or 'ceph'
storage_backend: 'swift'

# Storage disk size
storage_disk_mb_size: { swift: 761985, ceph: 914573, journal: 0 }

# Storage disk PD slot number
storage_disk_pd_slot: { swift: [6,7], ceph: [3,4,5,6], journal: [0] }
```

```
# Storage disk VD slot number
storage_disk_vd_slot: { swift: [238,239], ceph: [236,237,238,239], journal: [0] }

# Firmware version 'yes' or 'no' ???
firmware: { check: 'no', bios_version: '4.2.3c', cimc_version: '4.2(3e)' }

# ============================
# OSP Specific Variables
# ============================


# Timezone for overcloud nodes




timezone: 'Asia/Kolkata'




# Overcloud node count to deploy




node_count: { ctrl: 3, osdc: 0, cmpt: 11, mgmt: 2 }



local_network: {
   ip_type: 'v6',
   tenant_vlan_id: 1045,
   tenant_net_cidr: '240f:ppp:rr:1045::/64',
   tenant_alloc_pools_start: '240f:ppp:rr:1045::10',
   tenant_alloc_pools_end: '240f:ppp:rr:1045:ffff:ffff:ffff:fffe',

   storage_vlan_id: 1043,
   storage_net_cidr: '240f:ppp:rr:1043::/64',
   storage_alloc_pools_start: '240f:ppp:rr:1043::10',
   storage_alloc_pools_end: '240f:ppp:rr:1043:ffff:ffff:ffff:fffe',

   storage_mgmt_vlan_id: 1044,
   storage_mgmt_net_cidr: '240f:ppp:rr:1044::/64',
   storage_mgmt_alloc_pools_start: '240f:ppp:rr:1044::10',
   storage_mgmt_alloc_pools_end: '240f:ppp:rr:1044:ffff:ffff:ffff:fffe',

   internal_api_vlan_id: 1042,
   internal_api_net_cidr: '240f:ppp:rr:1042::/64',
   internal_api_alloc_pools_start: '240f:ppp:rr:1042::10',
   internal_api_alloc_pools_end: '240f:ppp:rr:1042:ffff:ffff:ffff:fffe'
}

# External VLAN and IP configs
external_network: {


   ip_type: 'v6',
```

```
   vlan_id: 1046,

   default_route: '2405:XXXX:YYYY:1055::1',

   network_cidr: '2405:XXXX:YYYY:1055::/64',

   alloc_pool_start: '2405:XXXX:YYYY:1055::100',

   alloc_pool_end: '2405:XXXX:YYYY:1055::200',

   horizon_ip: '2405:XXXX:YYYY:1055::107'

}

# Neutron mechanism driver 'ovs' or 'ovn'
neutron: {
   driver: 'ovs',
   dvr: false,
   datacenter_vlan_start: 1050,
   datacenter_vlan_end: 1070
}


# ============================
# OS Specific Variables
# ============================

# RHEL kernel version

kernelversion: '4.18.0-305.88.1.el8_4.x86_64'



# E810 ICE driver
ice_driver: { check: 'yes', version: 1.12.6, intel_aux_version: 1.0.1 }

# ENIC and FNIC verison
nic_version: { enic: '2.3.0.53', fnic: '1.6.0.53' }

# IPMI watchdog timer config
watchdog: { action: enabled, version: '2.0.31-3.el8.x86_64', timer: 250 }

# StorCLI raid management
storcliver: '007.2612.0000.0000-1.noarch'
```

```
# ============================
# Platform Specific Variables
# ============================

# Buffer pool size based on platform type
innodb_buffer_pool_size: 1610
# ###################################
# END - XR21 Specific Variables
# ###################################
```

언더클라우드 구축

Undercloud는 7단계로 실행 가능한 스크립트를 사용하여 구축됩니다. 이 경우 점프 호스트 역할을 하는 KVM 호스트에서 모든 단계를 실행해야 합니다.

단계	태그	설명	플레이북 YAML
1단계.	체크 파일	OSPD(Openstack Platform Director)에서 필요한 플레이북, 스크립트 및 RPM을 확인합니다.	osp16_published_playbooks_verify.yml
2단계.	겐포드바르	common_vars.yml(하드웨어, 소프트웨어, 네트워크 세부사항), hw_m6_vars.yaml(CPU, 메모리, 허브 페이지, 디스크, NIC 등), rhel_84_vars.yml(RHEL, 커널, ICE 드라이버, NIC 버전), pf_esc_vars.yml(Elastic Services Controller(ESC) 세부사항), osp_16_vars.yml(OSP 버전, 시간대, IP 유형, VLAN ID, IP, 중성자 세부사항) 등과 관련된 POD 특정 변수 파일을 생성합니다.	osp16_generate_pod_specific_vars.yml
3단계.	사전 구축	FQDN(Fully Qualified Domain Name), NTP(Network Time Protocol)를 구성하고 디렉터 노드의 모든 패키지를 업데이트합니다.	osp16_pre_undercloud_deploy.yml
4단계.	퍼스트부트	이전 컨피그레이션 및 패키지 설치 후 Undercloud 디렉터 노드의 첫 번째 재부팅을 수행합니다.	osp16_undercloud_deploy.yml

5단계.	ucdeploy	디렉터에 Undercloud 스택 설치	osp16_undercloud_tuning.yml
6단계.	주	디렉터 노드에서 BIOS CPU C-state 설정을 구성합니다.	osp16_cstate.yml
7단계.	세컨더부트	BIOS 변경 후 Undercloud 디렉터에서 두 번째 재부팅합니다.	해당 없음

이름이 osp16_auto_undercloud_deploy.yml인 파일은 단일 반복으로 실행할 수 있는 기본 실행 가능한 플레이북이지만, 구축 문제의 경우 쉽게 문제를 해결할 수 있도록 다른 태그를 사용하여 단계적으로 플레이북을 실행하는 것이 좋습니다.

<#root>

#

cd /home/stack/ansible/

ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=

TAG

For Ex –

#

ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=checkfiles
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=genpodvars
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=preucdeploy
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=firstreboot
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=ucdeploy
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=cstate
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=secondreboot

Note :-

Deployment Logs would be generated in “/home/stack/autologs” in Director-VM.

Post-Checks for Verification of Undercloud Deployment.
“stackrc” & “undercloud.conf” file must be generated in /home/stack folder.

```
# sudo podman ps -a
# source stackrc
# openstack stack list
# openstack stack show <stack-name> --fit
# openstack server list
# openstack network list
# openstack subnet list
```

오버클라우드 구축

Overcloud는 HA 모드에서 최소 3개의 컨트롤러와 1개의 컴퓨팅으로 구축됩니다. Overcloud는 17단계로 실행 가능한 스크립트를 사용하여 구축됩니다. 이 경우 점프 호스트 역할을 하는 Director-VM에서 모든 단계를 실행해야 합니다.

단계	태그	설명	플레이북 YAML
1단계.	겐포드바르	common_vars.yml(하드웨어, 소프트웨어, 네트워크 세부사항), hw_m6_vars.yaml(CPU, 메모리, 허브페이지, 디스크, NIC 등), rhel_84_vars.yml(RHEL, 커널, ICE 드라이버, NIC 버전), pf_esc_vars.yml(ESC 세부사항), osp_16_vars.yml(OSP 버전, 시간대, IP 유형, VLAN ID, IP, 중성자 세부사항) 등과 관련된 Overcloud에 대한 POD 특정 변수 파일을 생성합니다.	osp16_generate_pod_specific_vars.yml
2단계.	내장 경전	이전 단계에서 생성한 /var/common_vars.yml에서 Instackenv JSON 파일을 생성합니다. 디렉터는 노드 정의 템플릿을 필요로 하며, 이는 수동으로 생성됩니다. 이 파일 instackenv.json은 JSON 형식을 사용하며 노드에 대한 모든 하드웨어 및 전원 관리 세부 정보를 포함합니다. 또한 이 단계에서는 파일을 생성하기 전에 UCS 서버의 하드웨어 컨피그레이션을 검증합니다.	osp16_generate_instackenv.yml
3단계.	cimcvd	common_vars.yml, hw_m6_vars.yaml 및 rhel_84_vars.yml을 참조하는 각 서	osp16_cimc_vd_configure.yml

		버에 CIMC 설정 및 VD(가상 디스크)를 구성합니다.	
4단계.	사전 구축	이 단계에서는 오버클라우드를 구축하기 위한 모든 사전 요구 사항을 수행합니다. FQDN, NTP를 설정하고 모든 패키지를 업데이트하며, 구축용 경로로 이미지를 푸시합니다.	osp16_pre_overcloud_deploy.yml
5단계	가져오기 노드	이 단계에서는 서버 CPU, 메모리, NIC, 인터페이스 및 네트워크 스위치의 포트가 내향적입니다. 모든 컨트롤러 및 컴퓨터에 대해 연결된 네트워크 스위치에서 자체 검사가 수행됩니다.	osp16_import_ironic_nodes.yml
6단계.	속틀	컨트롤러 및 컴퓨터에 대한 사용자 지정 템플릿 파일을 생성합니다. 사용자 지정 템플릿에서 실행 중인 모든 서비스에 대한 컨트롤러 및 컴퓨팅 역할을 정의합니다. 또한 인증서, 경로 등을 적용하여 시스템을 강화합니다.	osp16_generate_custom_templates.yml
7단계.	옥구축	이 단계에서는 Openstack Overcloud 배포가 수행됩니다. RHOSP 구축을 위해 Red Hat에서 제공하는 deploy.sh를 실행합니다.	osp16_overcloud_deploy.yml
8단계	일반 목록	이 단계에서는 Provisioning IP, IPMI(CIMC) IP 및 자격 증명이 저장되고 컨트롤러와 매핑되는 Ansible에서 사용할 인벤토리 YML 파일이 생성되고 자동화를 위해 시스템이 로그인하고 추가 단계를 실행합니다.	osp16_build_inventory_v3.py
9단계.	오프라인네레포	/etc/yum.repo.d/offline.repo 파일에서 Overcloud Offline REPO를 구성하고 외부 네트워크를 통한 repo 서버를 가리킵니다.	osp16_config_offline_repo.yml
10단계.	울타리	Shoot The Other Node In The Head (a fencing technique in HA	osp16_config_fencing.yml

		clusters)(STONITH)를 사용하여 모든 컨트롤러 노드에서 펜싱을 구성합니다.	
11단계.	raidcache(레이드캐시)	모든 컨트롤러에 대한 RAID 캐시 설정을 구성하고 컴퓨터도 SWIFT 스토리지 설정을 구성합니다.	osp16_raid_cache_tuning.yml
12단계.	dnfupdate	모든 노드의 모든 패키지에 대해 DNF 업데이트를 실행합니다.	dnf_update_all_packages.yml
13단계.	세티플링크	이 단계에서는 EPDG(Evolved Packet Data Gateway) 내부 및 데이터 트래픽에 대해 SR-IOV 포트에 대한 신뢰 모드 제어가 활성화됩니다. SR-IOV 포트에 대한 지원은 중성자로 제공되며 VM이 SR-IOV 가상 기능을 통해 네트워크에 액세스할 수 있도록 합니다.	osp16_setIpLink.yml
14단계.	감시	이 단계에서는 디렉터 노드의 IPMI 설정이 대역 외 연결을 통해 모든 서버의 관리 작업에 대해 구성됩니다.	osp16_config_ipmi_watchdog.yml
15단계.	얼음 드라이버	PCI(Peripheral Component Interconnect) 카드용 인텔 E810 ICE 드라이버를 EPDG용 버전 1.12.6으로 업데이트하여 인텔 NIC 포트를 SR-IOV로 사용합니다.	osp16_ice_driver_install.yml
16단계.	재부팅합니다	이전 단계를 실행한 후 모든 Overcloud 노드를 재부팅합니다.	osp16_reboot_overcloud_hosts.yml
17단계.	입증되풀	RHOSP 구축 컨피그레이션 및 상태를 확인합니다.	osp16_rhosp_verify.yml

Overcloud 노드의 프로비저닝을 위해 Undercloud는 'overcloud-hardened-uefi-full.qcow2'를 사용합니다. 따라서 Overcloud 구축을 시작하기 전에 이미지를 언더클라우드/디렉터의 지정된 경로에 저장해야 합니다.

원격 사이트에서 Overcloud qcow2 파일을 복사합니다.

<#root>

```
# su - stack
# cd /home/stack
# mkdir deliverymedia
# cd deliverymedia
```

Copy overcloud-hardened-uefi-full.qcow2 to deliverymedia

```
# scp overcloud-hardened-uefi-full.qcow2 stack@[Director-IP]:/home/stack/deliverymedia
[stack@[stack@ Undercloud ~]$ cd /home/stack/ansible/
[stack@[stack@ Undercloud ansible]$ ansible-playbook osp16_auto_overcloud_deploy.yml -e podname=POD_NAM
```

For Ex –

```
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=genpodvars
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=geninstack
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=cimcvd
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=preocdeploy
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=importnodes
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=gentemplates
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=ocdeploy
```

Push & Update the rootCA.pem in all the Controllers & Computes

```
# for node in $(nova list | grep -i active | awk '{print $12}' | awk -F "=" '{print $2}' ); do scp -o S
```

Append the Director Entry in "/etc/hosts" file

```
# for node in $(nova list | grep -i active | awk '{print $12}' | awk -F "=" '{print $2}' ); do ssh -o S

# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=geninventory
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=offlinerepo
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=fencing
```

In case of Fencing Failures, please check the reachability of CIMC Subnet from Controllers

If CIMC Subnet is not pinging, Do add the static Route

```
# ip -6 route add <CIMC Subnet> via <Provisioning Subnet>
Ex: ip -6 route add 2405:XXXX:YYY:9999::/64 via 2405:XXXX:YYY:9999:1

# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=raidcache
```

```
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=dnfupdate
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=setiplink
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=watchdog
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=icedriver
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=reboot
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=verifyrhosp
```

구축 로그를 모니터링하려면 최신 로그 파일을 사용하십시오.

```
# tail -F </home/stack/autologs/osp16_auto_overcloud_deploy_*.log>
```

17단계를 모두 통과했는지 확인합니다.

검사 실패 => 수는 0이어야 합니다.

로그 => /home/stack/autologs/osp16_rhosp_verify.yml _20200703T042257.log

#==========

단계 | 태그 | 설명 | 플레이북

#==========

1단계 | genpodvars | POD 관련 변수 파일 생성 | osp16_generate_pod_specific_vars.yml -e podname=

2단계 | geninstack | Instackenv JSON 파일 생성 | osp16_generate_instackenv.yml -e podname=

3단계 | cimcvd | CIMC VD 구성 | osp16_cimc_vd_configure.yml

4단계 | 사전 구축 | Pre Overcloud Deploy 구성 | osp16_pre_overcloud_deploy.yml

5단계 | 가져오기 노드 | Openstack Baremetal Ironic 노드 가져오기 | osp16_import_ironic_nodes.yml

6단계 | 일반 서식 파일 | 사용자 지정 템플릿 생성 | osp16_generate_custom_templates.yml

7단계 | ocdeploy | Openstack Overcloud 구축 | osp16_overcloud_deploy.yml

8단계 | 일반 인벤토리 | 인벤토리 파일 생성 | osp16_build_inventory_v3.py —ipmipass

9단계 | offlinerepo | Overcloud Offline Repo from Offline TAR 파일 구성 | osp16_config_offline_repo.yml

10단계 | 펜싱 | 재부팅 전에 MOP 구축 후 - 펜싱 구성 | osp16_config_fencing.yml

11단계 | raidcache | 재부팅 전에 MOP 구축 후 - Raid 캐시 및 PR 조정 | osp16_raid_cache_tuning.yml

12단계 | dnfupdate | 재부팅 전에 MOP 구축 후 - Dnf 업데이트 패키지 | dnf_update_all_packages.yml

13단계 | setiplink | 재부팅 전에 MOP 배포 후 - VF IP 링크 트러스트 설정 | osp16_setIpLink.yml

14단계 | 감시 | 재부팅 전에 MOP 배포 후 - IPMI Watchdog 구성 | osp16_config_ipmi_watchdog.yml

15단계 | icedriver | 재부팅 전에 MOP 배포 후 - E810 ICE 드라이버 업데이트 | osp16_ice_driver_install.yml

16단계 | 재부팅 | 모든 Overcloud 노드 재부팅 | osp16_reboot_overcloud_hosts.yml

17단계 | verifyrhosp | RHOSP 구축 컨피그레이션 및 상태 확인 | osp16_rhosp_verify.yml -e podname=

#==========

연결 가능한 모든 호스트

==========

확인 완료 => 17

초과 검사 => 17

검사 실패 => 00

==========

전체 상태 => 통과 !!

==========

Overcloud를 성공적으로 구축한 후 Horizon 대시보드에 액세스할 수 있는지 확인합니다.

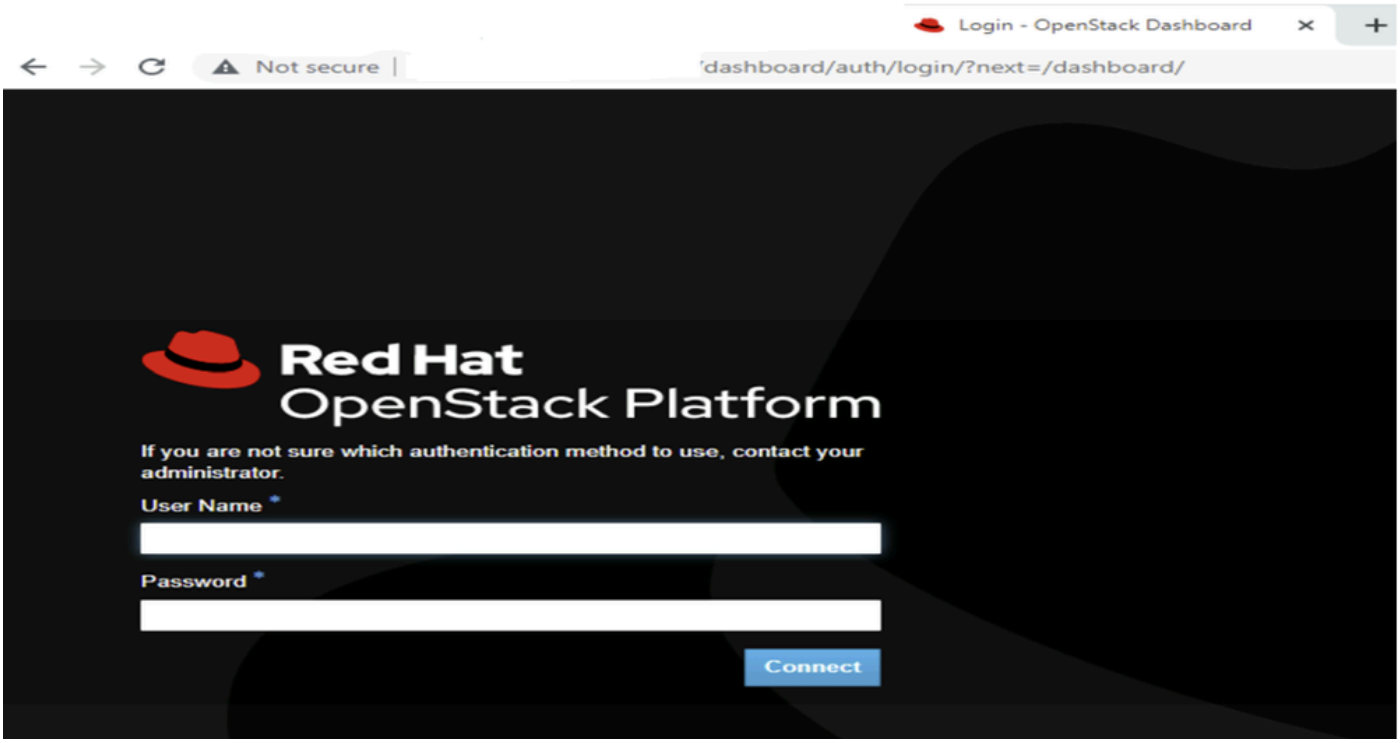
Horizon 대시보드 액세스

Horizon 대시보드 URL의 경우 'overcloudrc'에서 'OS_AUTH URL'을 사용합니다.

```
[stack@MUMBMUMBTCUDR201CO-ospd ~]$ cat MUMBMUMBTCUCL200COrc
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="}  /^OS_/ {print $1}' ); do unset $key ; done
export NOVA_VERSION=1.1
export COMPUTE_API_VERSION=1.1
export OS_USERNAME=admin
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_NO_CACHE=True
export OS_CLOUDNAME=MUMBMUMBTCUCL200CO
export no_proxy=',[                    ],[
export PYTHONWARNINGS='ignore:Certificate has no, ignore:A true SSLContext object is not available'
export OS_AUTH_TYPE=password
export OS_PASSWORD=openstack
export OS_AUTH_URL='http://[                    ]:5000'
export OS_IDENTITY_API_VERSION=3
export OS_COMPUTE_API_VERSION=2.latest
export OS_IMAGE_API_VERSION=2
export OS_VOLUME_API_VERSION=3
export OS_REGION_NAME=regionOne

# Add OS_CLOUDNAME to PS1
if [ -z "${CLOUDPROMPT_ENABLED:-}" ]; then
    export PS1=${PS1:-""}
    export PS1=\${OS_CLOUDNAME:+"(\$OS_CLOUDNAME)"}\ $PS1
    export CLOUDPROMPT_ENABLED=1
fi
```

Horizon 대시보드:



RHOSP 클러스터의 상태 확인

<#root>

```
### Check OpenStack Services Status ###


# openstack compute service list
# openstack network agent list
# openstack volume service list
# openstack orchestration service list
# openstack identity service list
# openstack endpoint list
# openstack server list
# openstack image list
```

요약

RHOSP 16.2 구축 설명서는 Red Hat의 검증된 툴과 방법론을 사용하여 확장 가능하고 프로덕션 환경에 적합한 OpenStack 클라우드 환경을 구축하기 위한 포괄적인 단계별 지침을 제공합니다. 이 가이드는 시스템 관리자 및 클라우드 설계자를 위해 제작되었으며 TripleO(OpenStack on OpenStack)를 기반으로 하는 OpenStack Director를 사용하여 RHOSP 16.2를 구축하는 데 중점을 두고 있습니다.

이 가이드에서는 다음을 비롯한 구축의 모든 주요 단계를 다룹니다.

- 인프라 계획 및 사전 요구 사항
- 환경 준비 및 네트워크 구성
- 언더클라우드 설치 및 컨피그레이션
- 오버클라우드 구축 및 구축 후 단계
- HA, 스토리지 및 서비스 확장 옵션

이 가이드는 에코시스템 통합 및 Red Hat의 지원을 통해 신뢰할 수 있는 엔터프라이즈급 클라우드 플랫폼을 원하는 팀에 꼭 필요합니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.